1. **NP-Completeness Problem 1:**

Let \( G = (V, E) \) be a directed graph and \( s, t \) be two vertices of \( G \). A hamiltonian path is a simple path from \( s \) to \( t \) that contains each vertex \( v \in V \) exactly once. Define

\[
HAMPATH = \{ (G, s, t) \mid G \text{ has a hamiltonian path from } s \text{ to } t \}
\]

Similarly, let \( G' = (V', E') \) be an undirected graph and \( s', t' \) be two vertices of \( G' \). A hamiltonian path is a simple path from \( s' \) to \( t' \) that contains each vertex \( v' \in V' \) exactly once. Define

\[
UHAMPATH = \{ (G', s', t') \mid G' \text{ has a hamiltonian path from } s' \text{ to } t' \}
\]

Assuming \( HAMPATH \in \text{NPC} \), prove that \( UHAMPATH \in \text{NPC} \).

2. **NP-Completeness Problems 2:**

Given an undirected graph \( G \), let

\[
SPATH = \{ (G, a, b, k) \mid G \text{ contains a simple path from } a \text{ to } b \text{ of length at most } k \}
\]

\[
LPATH = \{ (G, a, b, k) \mid G \text{ contains a simple path from } a \text{ to } b \text{ of length at least } k \}
\]

(a) Prove that \( SPATH \in \text{P} \).

(b) Prove that \( LPATH \in \text{NPC} \).

3. **NP-Completeness Problems 3:**

Consider the 2-SAT problem which is defined as:

\[
2\text{-SAT} = \{ \varphi \in \text{SAT} \mid \varphi \text{ is 2-CNF} \}
\]

(a) Is \( 2\text{-SAT} \in \text{NP} \)? Prove your conclusion.

(b) Is \( 2\text{-SAT} \in \text{P} \)? Prove your conclusion.

4. **Approximation Algorithms Problem 1:**

Let \( G = (V, E) \) be a weighted complete graph with \( n \) vertices. The travelling-salesman problem is to find the hamiltonian cycle of \( G \) with minimum cost. This problem is \text{NP-hard}.

Consider the following closest-point heuristic for building an approximate travelling-salesman tour. Begin with a trivial cycle consisting of a single arbitrarily chosen vertex. At each step, identify the vertex \( u \) that is not on the cycle but whose distance to any vertex on the cycle is minimum. Suppose that the vertex on the cycle that is nearest to \( u \) is vertex \( v \). Extend the cycle to include \( u \) by inserting \( u \) just after \( v \). Repeat until all vertices are on the cycle. Prove that this heuristic returns a tour whose total cost is not more than twice the cost of an optimal tour.
5. **Approximation Algorithms Problem 2:**

   Give an efficient greedy algorithm that finds an optimal vertex cover for a tree in linear time.